What is claimed is :

A system, and method for automating the development of multithreaded applications running on computing machines equipped with symmetrical multiple processors, and shared memory.

1) A system including a plurality of symmetrical processors, and a shared memory, operating under the control of an operating system and utilizing the runtime libraries of a programming language, called host language, comprising:

   a) Resource control programming (Rcp) runtime means, a translator, and a set of run time libraries, for providing translation and run time support to the application.

   b) Function means a sequence of instructions which accomplish a particular task.

   c) Invocation means a particular instance of execution, of said function.

   d) Element means a block of storage allocated in said shared memory.

   e) Queue means a container for a plurality of said elements and control structures for synchronizing access to said elements, whereby said elements contained in the queue are accessed by an unique identification number called element number.

   f) Queue array means a container for a plurality of said queues and control structures, for storing said queues, whereby said queues contained in the

queue array are accessed by an unique identification number called queue array number.

g) Virtual queue means a special kind of said queue which contains a reference to said queue or a combination of said queue array and said queue.

h) Rcp gate means a special function whose run time behavior is supplied by said Rcp run time libraries, and which comprises of zero or more said queues or said queue arrays on the input side, and zero or more queue arrays on the output side, and a control table called bind table, and control variables called inputs pending, inputs available, outputs available, outputs processed, number of assignments made, and next anticipated input identification number.

i) Node function means said function with a predefined signature, which comprises of zero or more said virtual queues defined on the input side, and zero or more said virtual queues defined on the output side, and has control structures for storing the runtime information of said invocations, and is associated with said Rcp gate.

j) Node function invocation means a particular instance of execution of said node function.

k) Producer means said node function which has one or more said virtual queues defined on the output side, or said Rcp gate which has one or more said queue arrays defined on the output side.

l) Consumer means said node function which has one or more said virtual queues defined on the input side, or said rcp gate which has one or more

said queues or said queue arrays defined on the input side.

m) Local ring means a control structure comprising of control information called bind info bits, and bind sequence number, which are used for synchronizing access, and assigning unique sequence numbers to data written to said queues of said queue arrays defined on the output side of said Rcp gate, such that whenever said queue arrays defined on the output side of the Rcp Gate are shared, by other said Rcp gates, the said local ring structure is shared, by all said Rcp Gates.

n) Ready state of said queue means that said producer has written data to the queue and has marked the queue as ready, for further processing by said consumers of the queue.

o) Not ready state of said queue means that the queue is available for output, and that said consumers of the queue, if any, are not currently using the queue, and that there is no data available for use in the queue.

p) Null state of said queue means that said producer has no data to write to the queue and has marked the queue as null, so that said consumers can avoid processing the queue.

q) Input Queue index means an index number, such that all said queues, identified by the index number, within said queue arrays, defined on the input side of said Rcp gate, are in said ready state.

r) Output Queue index means an index number, such that all said queues, identified by the index number, within said queue arrays, defined on the output side of said Rcp gate, are in said not ready state.

s) Bind Sequence number means a sequential number assigned by said Rcp gate to said queues, at said output queue index.

t) Queue disposition means said queue array, to which said queue under consideration belonging to another said queue array will be copied, when the queue is released by all of its said consumers.

u) Worker means a thread, and control structures for controlling said thread, having an unique identification number called worker number, within said frame.

v) Rcp resource means any of said queues, said queue arrays, said virtual queues, said node functions, said Rcp gates, said local rings, and said workers.

w) Frame means a partition within the application process, containing control tables for storing said Rcp resource definitions and their runtime statuses, and having an unique identification number called, frame number.

x) Run identification or Run_id means a control structure received by said node function invocation when it is invoked at run time, and which is comprised of said frame number and said worker number.

y) Resource control programming (Rcp) Statements means, a high level language mechanism for defining, accessing and controlling said Rcp resources.

z) Dispatcher means said worker within said frame, which acquired a lock contained in said control structures of said frame, whereby said worker can

assign, said node function invocations waiting for execution, to itself, and other said workers which are idle, within said frame.

2) A method of automating the development of multithread applications in a computing system, containing a plurality of symmetrical multiple processors, comprises the steps of :

a) Defining and initializing said Rcp resources, as per the requirements of the application.

b) Specifying said Rcp Statements in the source files of the application, for accessing, modifying and controlling said Rcp resources, defined for the application.

c) Translating said Rcp statements into host language statements or internal control structures, and storing said internal control structures in a load image file.

d) Generating a function called Rcp_Init function, for initializing said Rcp resources defined for the application.

e) Building an executable module for the application, by compiling and optionally linking the translated source files and said Rcp_init function generated by said translator.

f) Invoking said Rcp Runtime by issuing the Rcp statement "Run Pgm" from the application.

g) Waiting for all said frames to terminate

3) The method in Claim 2, further comprises of :

a) Initializing said Rcp Runtime environment, and creating said frames, and said workers within each of said frames, and executing said Rcp_Init function generated by said translator, whereby the function pointers to said node functions are acquired by the Rcp runtime library.

b) Determining said node function invocations which can be executed, within each said frame, and executing said node functions within each said frame, until a Stop, abend, or Idle event is generated within each said frame.

4) The method in Claim 3, further comprises of :

a) Performing an activity called binding whereby a complete set of said queues, identified by said input queue index on the input side of said Rcp gate and a complete set of said queues identified by said output queue index on the output side of said rcp gate are determined and stored in the control structures of said Rcp gate. The queue indices on the input and output side of the Rcp gate are collectively called a binding. This activity is carried out for each said Rcp gate, in each said frame, by said dispatcher of said frame.

b) Determining if said Rcp gate is running efficiently, and selecting said node function invocation, from a plurality of said node function invocations waiting for execution.

c) Performing an activity called rebind, whereby said Rcp gate associates said binding, with said node function invocation.

d) Assigning a worker to said node function invocation bound to said queues.

e) Executing said node function invocation, which contains host language statements, and returning back to said rcp runtime.

5) The method in Claim 4, performing an activity called binding, further comprises the steps of :

a) Terminating said Rcp Gate when said input queues contained in said input queue arrays are all in said not ready state, and said producers for at least one of the queue arrays have terminated

b) Determining for each valid value of said input queue index of said Rcp gate, if said bind sequence number of said queues, is greater than or equal to said next anticipated input identification number, stored in said control structures of said Rcp gate.

c) Storing said bind sequence number, and said input index, determined above, in said "bind table", of said Rcp gate, at a location in said bind table, obtained by hashing said bind sequence number with the size of said bind table.

d) Determining if any said queues identified by said input queue index are marked as null, and setting an internal flag called null flag in said bind table, where the bind table entry is identified by said bind sequence number of the queues identified by the input queue index.

e) Determining if there are any gaps in said bind sequence numbers, stored in said bind table, and incrementing said inputs pending counter of said Rcp gate with number of inputs after the first gap in said bind sequence

numbers, and incrementing said inputs available counter of said Rcp gate, with number of inputs without any gaps in said bind sequence numbers.

f) Determining the next valid value of said output queue index of said Rcp gate, and checking that the corresponding bind info bit stored in said local ring is zero, and when these conditions are met, said output queue index of said Rcp gate, and said next bind sequence number of said local ring are stored in said bind table, at the location identified by an internal index, which sequentially traverses said bind table. The next bind sequence number of said local ring is incremented by 1. The corresponding bind info bit of the local ring is set to 1, and said outputs available counter of said Rcp gate is incremented. The said local ring is accessed in a thread safe manner.

g) Determining if said inputs available counter and said outputs available counter of said Rcp gate are positive, and returning a special return code, to signal failure of the bind activity, if any of said counters are zero.

6) The method in claim 4, determining if said Rcp gate is running efficiently, further comprises of :

Determining the efficiency of said Rcp gate, by the formula

Rcp Gate efficiency = (Num of outputs processed by the Rcp Gate * 100) /
(Num of worker assignments for all the invocations of
the node functions *
min(min(capacity of input queue arrays),
min(capacity of output queue arrays)));

If number of worker assignments for all node function invocations, determined by the number of assignments made value of said Rcp gate, is zero, then said Rcp gate efficiency is set to 100%. If said producers have terminated for any one of said queue arrays on input side of said Rcp gate, then said Rcp gate efficiency is set to 100%.

If said Rcp gate efficieny is below 25%, and if said inputs available and said outputs available counters of said Rcp gate are less than 25% of the minimum capacity of said queue arrays of said Rcp gate, further processing is bypassed, since said Rcp gate is operating poorly, which means that more data should be accumulated before said node function invocations of said Rcp gate are started.

If said Rcp gate efficiency is above 75%, and if said inputs available and said outputs available counters of said Rcp gate are greater than 75% of the minimum capacity of said queue arrays of said Rcp gate, then another invocation of said node function can be started, if said node function invocation is available for execution.

7) The method in Claim 4, performing an activity called rebind, further comprises the steps of :

a) Selecting said bind table entry, in a thread safe manner, using an index called "Rebind index" stored in the control structures of said Rcp gate, which traverses said bind table entries sequentially, and wraps around after the last entry.

b) Skipping said bind table entry if the null flag of the entry is set to 1.

c) Copying said input queue index, said output queue index, and said bind sequence number from said bind table entry identified by said Rebind index to the control structures of said node function invocation,.

d) Marking said bind table entry identified by said rebind index as "Rebind complete".

8) The method in Claim 4, assigning a worker to said node function, comprises of :

Marking said node function invocation as Waiting for execution, so that it will be dispatched for execution by said dispatcher, when said worker becomes available.

9) The method in claim 4, executing said node function invocation, and returning back to said rcp runtime, further comprises the following steps :

a) Executing the host language statements.

b) Optionally reading said queues on the input side of said node function by executing Rcp statement "Read Queue".

c) Optionally writing data to said queues on the output side of said node function, and setting it to said ready state by executing Rcp statement "Add Queue", whereby said bind sequence number contained in the control structures of said node function invocation, is copied to the control structures of said output queue array, when said queue belonging to said

queue array is set to said ready state.

d) Terminating said node function invocation by executing the rcp statement "Release queues", when said Rcp gate associated with said node function has no input queue arrays, defined on its input side.

e) Optionally executing a Rcp statement "Rebind queues", to acquire another set of said queues, and re executing the host language statements of said node function., until said Rcp statement "Rebind queues" returns a special return code to signal failure, whereby control is returned back to the Rcp run time.

10) The method in claim 9, optionally executing said Rcp statement "Rebind queues", to acquire another set of said input and output queues, further comprises the steps of :

a) Performing an operation called unbind, whereby said input and output queues bound to said node function invocation, are released.

11) The method in calim 10, performing an operation called unbind, further comprises the steps of :

a) Releasing said queues bound to said node function invocation on the input side, whereby for each said queue, a control field in said control structures of said queue, containing current count of said consumers, is decremented by 1, and when the current count of said consumers drops down to zero, said queue control structures are reset, and said queue is set to said not ready state.

b) Releasing said queues bound to said node function invocation on the output side, whereby said bind info bit of said local ring, corresponding to said output queue index contained in the control structures of said node function invocation is set to zero, in a thread safe manner.